Remarks

The claimed invention concerns a recursive application of a database query to a database catalog to identify dependencies for database code modules. Code modules in PL/SQL can reside in separate library units that reference one another. There are five main types of library units with which a de-bugger may interact:

1. packages, which are a collection of stored procedures and/or functions;
2. procedures;
3. functions;
4. triggers; and
5. stored types.

Packages can be implemented as two distinct code objects, a package specification and a package body. Similarly, types can be stored as specifications and bodies. The application claims systems, methods and computer readable media for automatically generating complete dependency information for such DBMS stored code objects. The systems and methods include recursively querying a database catalog to find dependencies. In the prior art, only one level (e.g., direct) dependencies are retrieved from a database catalog. Furthermore, in the database field, stored procedures are not compiled and linked together to find dependencies because dependency information is directly available from the database catalog and thus, the prior art concerning compiler methods are inapplicable to the claims in the application.

Claim Rejections Under 35 U.S.C. §103

This introductory section first discusses the law related to Section 103, and how the Office Action does not satisfy the law.

The four criteria that underlie a proper application of Section 103 are:

1. a determination of the scope and content of the prior art;
2. a determination of the differences between the prior art and the claims at issue;
3. a determination of the level of ordinary skill in the pertinent art; and
4. a determination of which, if any, secondary considerations are relevant and the effect of those secondary considerations.

Mitsubishi Elec. Corp. v. Ampex Corp., 190 F.3d 1300, 1308-09 (Fed. Cir. 1999) cert. denied, 120 S.Ct. 1556 (2000). The Office Action has not determined the scope and content of

the prior art, the differences between the prior art and the claims at issue, and the level of ordinary skill in the art. Thus, Section 103 has not been properly applied.

MPEP §2142 reads "to establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves, or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, **the prior art reference** (or references when combined) **must teach or suggest all the claim limitations**. The teaching or suggestion to make the claim or a combination and reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure." (emphasis added) *In re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991). Similarly, MPEP §2143.03 reads "to establish *prima facie* obviousness of a claimed invention, all the claimed limitations must be taught or suggested by the prior art." *In re Royka,* 490 F.2d 981 (CCPA 1974). Note that MPEP §2143.03 reads "if an independent claim is not obvious under 35 U.S.C. §103, then any claim depending therefrom is not obvious." *In re Fine*, 837 F.2d 1071 (Fed. Cir. 1988). The references cited in the Office Action do not disclose or suggest all the elements claimed in the rejected claims and, thus a *prima facie* case of obviousness cannot be made.

Before references can be combined, there must be actual evidence of a suggestion, teaching, or motivation to combine such prior art references. *Ecolochem Inc. v. Southern Cal. Edison Co.,* 227 F.3d 1361, 1371-72 (Fed. Cir. 2000). There is no such suggestion, teaching, or motivation to combine any of the references cited in the Office Action. Furthermore, an Examiner should not engage in "hind sight" reconstruction. *In re Kotzab,* 217 F.3d 1365, 1369-72 (Fed. Cir. 2000), *In re Dembiczak,* 175 F.3d 994, 998-1000 (Fed. Cir. 1999). Clearly, piecing together unrelated references that may include one or more elements of the present application is an attempt at hindsight reconstruction. "Our case law makes clear that the best defense against the subtle but powerful attraction of hindsight-based obviousness analysis is rigorous application of the requirement for showing of the teaching or motivation to combine prior art references" *Dembiczak*, p. 999. Here, because there was no application of the requirement for showing a teaching or motivation to combine, the Office Action succumbs to the powerful attraction. "Combining prior art references without evidence of such a suggestion, teaching, or motivation simply takes the inventor's disclosure as a blue print for piecing together the prior art to defeat

patentability - the essence of hindsight." *Id.* Here, while the blueprint approach has been followed, the Office Action simply recites a disconnected parts list that does not teach or suggest the elements claimed in the rejected claims.

"Identification in the prior art of each individual part claimed is insufficient to defeat patentability of the whole claimed invention. Rather, to establish obviousness based on a combination of the elements disclosed in the prior art, there must be some motivation, suggestion or teaching of the desirability of making the specific combination that was made by the applicant." *In re Kotzab*, 217 F.3d 1365, 1369-70 (Fed. Cir. 2000) (internal citations omitted). Not only does the Office Action merely provide a disconnected parts list, the list is incomplete, missing claimed elements in each and every rejected claim.

"A rejection cannot be predicated on the mere identification [in a reference] of individual components of claimed limitations. Rather, particular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed." *Kotzab*, p. 1371. Given that the Office Action provides no findings, let alone particular findings, applicants submit that the rejections cannot stand.

Claims 9 and 15

Claims 9 and 15 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Sankaran, et al. (U.S. Patent No. 5,832,484) in view of Morgenstern (U.S. Patent No. 5,970,490).

Claim 9 reads:

> A method of generating a basic dependency tree of a code object that does not take into consideration dependencies on triggers and on implementations of object oriented code objects, the method comprising the steps of:
>
> querying a database catalog for direct dependencies of a code object and then for each dependency found, doing the query recursively until all basic dependencies are generated into a dependency tree.

Sankaran concerns a client/server database system with methods for providing parallel lock management to facilitate scaling into symmetric multiprocessing (SMP) environments.

"Accordingly, the implementation of parallel lock management requires improved deadlock searching methodology." Col. 12, ll. 65-67. Thus, Sankaran is clearly not related to dependencies of code objects and database catalogs and there is no teaching, motivation or suggestion to combine Sankaran with Morgenstern. Furthermore, while Sankaran employs dependency graphs to facilitate detecting deadlocks, Sankaran uses a conventional dependency graph in a conventional way. It does not disclose querying a database catalog or generating direct dependencies for objects stored in a database by using data stored in the database catalog. Therefore, Sankaran omits elements of claim 9 including, querying a database catalog for direct dependencies and applying the query recursively until dependencies are generated into a dependency tree.

Morgenstern does not overcome the shortcomings of Sankaran. Morgenstern concerns an integration platform for heterogeneous databases. It claims methods for processing heterogeneous data that employ an interoperability assistant module with specifications for transforming data into a common intermediate representation of the data. Morgenstern produces an "information bridge" unrelated to debugging stored procedures in databases. Morgenstern is clearly not related to dependencies of code objects and database catalogs. Therefore, since neither Sankaran nor Morgenstern concern stored procedures in databases, let alone debugging such procedures with the help of complete dependency trees generated by recursively querying a database catalog, nothing in either reference would lead one skilled in the art to combine these references in the manner proposed in the Office Action. Furthermore, the proposed combination of Sankaran and Morgenstern omits the claimed elements of querying a database catalog for direct dependencies and employing a query recursively until basic dependencies are generated into a dependency tree. Since neither Sankaran nor Morgenstern, alone or in combination, teach or suggest all the claimed limitations in claim 9, a *prima facie* case of obviousness cannot be made out for claim 9 based on Sankaran and Morgenstern. Therefore, applicants respectfully submit that claim 9 is patentable.

Concerning claim 15, which depends from claim 9, the combination of Sankaran and Morgenstern does not teach or suggest the claim limitations in claim 9 and therefore since claim 9 is not obvious, claim 15 is not obvious. Furthermore, in addition to the missing elements in claim 9, from which claim 15 depends, the additional limitation in claim 15 of "using the generated dependency tree to identify cyclic dependencies among database code objects" is

neither taught nor suggested by either Sankaran or Morgenstern. For this additional reason, claim 15 is not obvious. Therefore, applicants respectfully request the withdrawal of this rejection.

Claims 10, 11, 13 and 16

Claims 10, 11, 13, and 16 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Sankaran in view of Morgenstern, and further in view of McKeeman (U.S. Patent No. 5,325,531).

Concerning claim 10, which depends from claim 9, as discussed above the combination of Sankaran and Morgenstern neither discloses the elements claimed in claim 9 nor provides any motivation or suggestion to combine the references. McKeeman does nothing to overcome the shortcomings of Sankaran and Morgenstern, similarly omitting the claimed elements of querying a database catalog for direct dependencies and recursively querying until all basic dependencies are generated into a dependency tree. Therefore, neither Sankaran, Morgenstern nor McKeeman, alone or in combination, teach or suggest the claim limitations in claim 10. Therefore, applicants respectfully request that the rejection of claim 10 be withdrawn.

Additionally, McKeeman does not provide the missing motivation to combine. McKeeman concerns an incremental compiler. McKeeman employs a dependency analysis that is performed incrementally, which is analogous to the cited prior art. McKeeman provides a programming environment designed to enhance speed and productivity of software development, "particularly a method for substantially decreasing the time required for recompilation and relinking in the edit-compile-link-run cycle of the software development process." Col. 1, ll. 52-55. Thus, it is likely that the invention claimed in McKeeman would suffer from the very problems the present application tries to solve (e.g., having to resolve dependencies by conventional linking). The Office Action provides no rationale for why Sankaran, Morgenstern and/or McKeeman should be combined in the proposed manner. Furthermore, McKeeman does not disclose the additional claim limitation of claim 10 of using a generated dependency tree to compile code objects in debug mode as part of a database code object debugging tool and thus does not render claim 10 obvious. For these additional reasons, applicants respectfully request that this rejection be withdrawn.

Concerning claim 11, which depends from claim 9, the shortcomings of Sankaran and Morgenstern concerning claim 9 (no motivation to combine, missing claimed elements) are not overcome through the addition of McKeeman. Additionally, McKeeman does not disclose the additional element claimed in claim 11 of employing a dependency tree to identify calling paths in a database code coverage tool. McKeeman is silent concerning paths and database code coverage tools. McKeeman merely discloses an incremental compiler that includes an adaptation to a traditional edit-compile-link-run method whereby a compile or link is ended as soon as a first error is noticed in the compiling or linking. This is unrelated to the elements claimed in claims 9 and 11. While McKeeman stores source text in a buffer and only recompiles code that is changed, it is silent concerning generating dependencies for stored database procedures, further evidence of its inapplicability to this application. Therefore, for the reasons stated above (missing elements, no motivation to combine) claim 11 is not obvious and applicants respectfully submit that claim 11 is patentable.

Concerning claim 13, which depends from claim 9, in addition to McKeeman not overcoming the shortcomings of Sankaran and Morgenstern (missing claimed elements, no motivation to combine), Morgenstern does not disclose the additional claim limitation in claim 13 of using a generated dependency tree in a database code object testing tool. Thus, since neither Sankaran, Morgenstern, nor McKeeman teach or suggest a database code object testing tool, the proposed combination of references does not make out a *prima facie* case of obviousness. In McKeeman, col. 2, ll. 34-60, and col. 6, ll. 1-12, McKeeman discloses a method for efficiently generating object code. This is employed to reduce the time for the traditional edit-compile-link-run cycle. It does not discuss discovering dependencies or employing discovered dependencies in a database code object testing tool. While McKeeman discloses saving "debugged object code", Col. 6, ll. 5-6, claim 13 concerns database code objects. Object code is not the same thing as a code object, and would not lead one skilled in the art to combine the references in the proposed manner. Thus, for this additional reason, applicants submit that this claim is patentable.

Concerning claim 16, which depends from claim 9, McKeeman does not overcome the shortcomings of Sankaran and Morgenstern concerning claim 9 (missing claim elements, no motivation to combine). Furthermore, McKeeman does not disclose the additional element claimed in claim 16 of using a generated dependency tree in a dependency graph presentation

tool. The incremental compiler described in McKeeman does not display a dependency graph. It is simply an incremental compiler that displays run time and/or link/compile errors. Similarly, neither Sankaran nor Morgenstern teach or suggest displaying a dependency graph, and thus, claim 16 is not obvious in light of the combination of such references. Therefore, applicants respectfully submit that this claim is patentable.

Claim 12

Claim 12 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Sankaran in view of Morgenstern and further in view of Jagannathan (U.S. Patent No. 5,692,193). Concerning claim 12, which depends from claim 9, Jagannathan does not overcome the shortcomings of Sankaran and Morgenstern with respect to claim 9 (missing claimed elements, no motivation to combine). Furthermore, Jagannathan does not disclose the additional claimed element in claim 12 of using a dependency tree generated from recursive queries on a database catalog to identify call paths in a database code object profiling tool. Jagannathan concerns a software architecture for controlling highly parallel computer systems. While Jagannathan reveals threads that contain genealogy information (e.g., parents, siblings, children), no provision is made in Jagannathan for recursively discovering further dependencies. Thus, Jagannathan is an example of prior art discussed in the background of the present application that may acquire a first level of dependencies but which makes no provision for recursively querying a database catalog for the complete collection of further dependencies. Thus, Jagannathan does not provide the missing motivation to combine the references in the proposed manner. The Office Action is silent with respect to such teaching, motivation, or suggestion, and thus, the rejection based on this combination should be withdrawn. Additionally, Jagannathan in combination with Sankaran and Morgenstern does not include the claimed elements of claim 12 and/or claim 9, and thus does not render claim 12 obvious. Therefore, applicants respectfully submit that this claim is patentable.

Claim 14

Claim 14 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Sankaran in view of Morgenstern and further in view of Eto (U.S. Patent No. 5,561,763). Concerning claim 14, which depends from claim 9, as described above, Sankaran and Morgenstern lack the claimed elements of querying a database catalog for direct dependencies and recursively querying until basic dependencies are generated into a complete dependency tree. Eto does not overcome the shortcomings of Sankaran and Morgenstern with respect to the missing claimed elements of claim 9. Thus, claim 14, which depends from claim 9, is not obvious. Furthermore, neither Sankaran, Morgenstern, nor Eto disclose the additional element claimed in claim 14 of identifying invalid dependent objects. Eto concerns discovering whether a transaction is invalid. As employed in Eto, a transaction concerns a data manipulation language (DML) statement. Eto facilitates determining whether a particular DML statement in a transaction will work. Eto does not facilitate determining whether a dependent object is valid, which is the additional element claimed in claim 14. Therefore, in addition to not disclosing the claim limitations of claim 9, the combination of Eto, Sankaran and Morgenstern does not disclose the additional limitation of identifying invalid dependent objects by employing a dependency tree generated through recursive queries to a database catalog. Therefore, the combination of references does not render claim 14 obvious and applicants respectfully request that this rejection be withdrawn. Additionally, there is no teaching, motivation, or suggestion to combine the references in the manner proposed and thus, for this additional reason, applicants respectfully request the withdrawal of this rejection.

Claims 17, 18, 20, 22, 23, 25, and 27-29

Claims 17, 18, 20, 22, 23, 25, and 27-29 stand rejected under 35 U.S.C. §103(a) as being unpatentable over McKeeman in view of Doo (U.S. Patent No. 5,926,819). The combination of McKeeman and Doo, for which there is no motivation, teaching, or suggestion to combine provided in the Office Action, does not disclose elements claimed in independent claims 17, 22 or 27. McKeeman discloses an incremental compiler. The Office Action asserts that McKeeman discloses a "recursion parser". Page 9, line 1. The present application does not relate to a "recursion parser". It is dubious whether a "recursion parser" even exists and thus, whether McKeeman discloses a "recursion parser" is irrelevant. A canonical parse of a programming language that has a context free grammar is unrelated to database catalog querying using a

recursive technique to generate dependencies and thus there is no reason one skilled in the art would consider McKeeman in combination with Doo to attempt to produce the elements claimed in the present application.

Similarly, the Office Action asserts that Doo discloses that databases perform row replication through triggers. Performing row replication through in-line trigger operation is completely unrelated to generating dependency information including dependencies of code objects on database triggers. Doo does not describe parsing code objects in a dependency graph to identify DML statements that fire triggers thereby identifying dependencies on triggers. Therefore, like McKeeman, there is no reason one skilled in the art would consider Doo in any combination that attempts to produce the elements claimed in the application. Thus, not only is there no reason to combine the references, but the combination of McKeeman and Doo does not teach or suggest the claimed elements of employing a recursive algorithm for querying a database catalog for direct dependencies of a code object or using a parser on code objects to identify DML statements that fire triggers thereby identifying dependencies on triggers. Furthermore, neither McKeeman nor Doo teach or suggest applying a recursive query against a database catalog for triggers to incorporate trigger dependencies into a dependency graph. Therefore, the combination of McKeeman and Doo does not render claim 17 obvious and applicants respectfully submit that the claim is patentable.

Concerning claim 18, which depends from claim 17, claim 17 is not obvious (missing claimed elements, no motivation to combine) and therefore neither is claim 18. Furthermore, contrary to the assertion in the Office Action that McKeeman discloses paths in a database coverage tool, McKeeman does not discuss working with database objects, but rather discusses working with an incremental compiler. McKeeman is silent concerning identifying calling paths in a database code coverage tool and thus in addition to the shortcomings with respect to claim 17 it is deficient concerning the additional claimed element in claim 18. Therefore, claim 18 is not obvious in light of McKeeman and Doo and applicants respectfully request that the rejection be withdrawn.

Concerning claim 20, which depends from claim 17, claim 17 is not obvious (missing claimed element, no motivation to combine) and therefore claim 20 is not obvious. Furthermore, neither McKeeman nor Doo teach or suggest the additional element claimed in claim 20 of using

a generated dependency information in a database code object testing tool. Therefore, for this additional reason, applicants respectfully request that this rejection be withdrawn.

Concerning claim 22, an independent claim, the Office Action asserts that claim 22 is obvious in light of the combination of McKeeman and Doo. McKeeman discloses an incremental compiler that performs a recursive descent of a parse tree. McKeeman does not disclose querying a database catalog for dependencies, particularly not for dependencies associated with implementations of code objects in a database. As discussed earlier, a code object can have two pieces, a specification and an implementation. Claim 22 applies the recursive algorithm to facilitate incorporating dependencies of implementations of code objects in the database. Neither McKeeman nor Doo identify object oriented code objects in a database, let alone disclose performing a recursive database query for one of the component parts of an object oriented code object, the implementation of the object oriented code. Therefore, since the combination of McKeeman and Doo does not disclose the elements claimed in claim 22, particularly applying a recursive algorithm that queries a database for dependency information and then recursively seeks dependencies for the implementations of code objects, the combination of McKeeman and Doo does not render claim 22 obvious. Therefore, applicants respectfully request that the rejection be withdrawn. Additionally, neither the Office Action nor either of the references provide a teaching, motivation or suggestion to combine the references in the proposed manner. Thus, for this additional reason, applicants request withdrawal of this rejection.

Concerning claim 23, which depends from claim 22, claim 22 is not obvious (missing claimed element, no motivation to combine) and therefore claim 23 is not obvious. Claim 23 claims the additional element of using a generated dependency information in a direct dependency graph to identify calling paths in a database code coverage tool. Neither McKeeman nor Doo discuss, teach, or suggest employing dependency information in the context of a database code coverage tool and therefore do not render claim 23 obvious. Therefore, for this additional reason, applicants respectfully request that this rejection be withdrawn.

Concerning claim 25, which depends from claim 22, claim 22 is not obvious (missing claimed elements, no motivation to combine) and thus claim 25 is not obvious. Furthermore, neither McKeeman nor Doo disclose the additional claimed element of a database code object

testing tool. Thus, in addition to the shortcomings related to claim 22, the combination of references suffers from the further shortcoming of not teaching or suggesting the additional claimed element of claim 25. Therefore, claim 25 is not obvious and applicants respectfully request that this rejection be withdrawn.

Concerning claim 27, an independent claim, the Office Action asserts that the combination of McKeeman and Doo renders claim 27 obvious. While Doo discloses a conventional hardware environment for a computing system, neither McKeeman nor Doo disclose the claimed elements of stored object oriented code objects, stored specifications of packages, stored implementations of packages, stored specifications of types, stored implementations of types and stored triggers. Thus, while the combination of McKeeman and Doo may disclose a computing system on which an incremental compiler may be implemented, the combination of references does not disclose a system for identifying dependencies that generates a dependency graph that has dependency information for code objects, specifications of packages, implementation of packages, specifications of types, implementation of types, triggers, and dependencies of triggers. Therefore, the combination of references does not teach or suggest all the claim limitations in claim 27 and therefore does not render claim 27 obvious. Therefore, applicants respectfully request that this rejection be withdrawn. Furthermore, there is no teaching, motivation or suggestion to combine the references in the proposed manner provided in either the Office Action or the references. For this additional reason, applicants respectfully request that the rejection be withdrawn.

Concerning claim 28, an independent claim, while Doo discloses a computer having a processor and so on, neither McKeeman nor Doo disclose the claimed elements of stored representations of object oriented code objects, specifications of packages, implementations of packages, specifications of type, implementation of types, and triggers. Furthermore, neither McKeeman nor Doo disclose using a recursive code mechanism for generating a dependency graph. Thus, it is impossible for McKeeman or Doo to teach or suggest a dependency graph that has a data structure with entries containing representations of dependent code objects, specifications of packages, implementations of packages, specifications of types, implementations of types, triggers, and dependencies of triggers as claimed in claim 28. Therefore, the combination of McKeeman and Doo does not render claim 28 obvious and applicants respectfully request that this rejection be withdrawn. Furthermore, there is no

teaching, motivation or suggestion to combine the references in the proposed manner provided in either the Office Action or the references. For this additional reason, applicants respectfully request that the rejection be withdrawn.

Concerning claim 29, an independent claim that claims a computer program product embedded on a computer readable medium, the combination of McKeeman and Doo does not disclose all the claimed elements in claim 29. For example, neither McKeeman nor Doo disclose a recursive code mechanism for generating a dependency graph of a target database code object. Furthermore, neither of the references discloses a dependency graph with a data structure storing entries containing representations of dependent code objects, specifications of packages, implementations of packages, triggers, and dependencies of triggers. Therefore, since neither reference discloses building such a dependency graph, it impossible for either reference to teach or suggest using such a dependency graph to debug a target database code object. Therefore, the combination of references does not teach or suggest all the elements of claim 29 and therefore does not render claim 29 obvious. Therefore, applicants respectfully request that this rejection be withdrawn. Furthermore, there is no teaching, motivation or suggestion to combine the references in the proposed manner provided in either the Office Action or the references. For this additional reason, applicants respectfully request that the rejection be withdrawn.

Additionally, it is to be noted that claim 29 claims a computer readable medium. Neither McKeeman nor Doo disclose a computer readable medium on which a system and/or method for use in debugging a target database code object is stored. Therefore, for this additional reason, claim 29 is not obvious.

Claims 19 and 24

Claims 19 and 24 stand rejected under 35 U.S.C. §103(a) as being unpatentable over McKeeman in view of Doo and further in view of Jagannathan. Claim 19 depends from claim 17, which as discussed above is not obvious (missing claimed elements, no motivation to combine). Therefore, claim 19 is not obvious. Similarly, claim 24 depends from claim 22, which as discussed above is not obvious (missing claimed elements, no motivation to combine), and therefore claim 24 is not obvious.

In addition to the combination of references not disclosing the claimed elements in claim 17, from which claim 19 depends, none of the references teach or suggest the additional claimed

element of using a generated dependency information to identify calling paths in a database code object profiling tool as claimed in claim 19. While Jagannathan discusses generating and/or retrieving dependency (genealogy) information for one level of a dependency hierarchy, neither McKeeman, Doo, nor Jagannathan teach or suggest recursively querying a database catalog for dependencies for either triggers or implementations. The purported disclosure of a database code object profiling tool in Jagannathan actually discusses monitoring "the dynamic unfolding of a process tree." Col. 11, ll. 30. While monitoring the dynamic unfolding of a process tree is important to software for controlling highly parallel computer systems as claimed in Jagannathan, it is not a database code object profiling tool as claimed in claim 19. Furthermore, there is no teaching, motivation or suggestion to combine the references in the proposed manner provided in either the Office Action or the references. For this additional reason, applicants respectfully request that the rejection be withdrawn.

Concerning claim 24, which depends from claim 22, the combination of McKeeman, Doo, and Jagannathan does not teach or suggest all the claimed elements in claim 22. Additionally, the combination of references does not teach or suggest the additional claimed element in claim 24 of using the generated dependency information in the direct dependency graph to identify calling paths in a database code object profiling tool. While Jagannathan discloses accessing genealogy information, it does not disclose recursively querying a database catalog for dependencies for either triggers or implementations. Therefore, the combination of references does not render claim 24 obvious and applicants respectfully request that this rejection be withdrawn. Furthermore, none of the references recites incorporating dependencies for the implementations of object oriented code objects to a dependency graph. Therefore, for this additional reason, claim 24, which depends from claim 22, is not obvious. Furthermore, there is no teaching, motivation or suggestion to combine the references in the proposed manner provided in either the Office Action or the references. For this additional reason, applicants respectfully request that the rejection be withdrawn.

Claims 21 and 26

Claims 21 and 26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over McKeeman in view of Doo and further in view of Eto. Claim 21 depends from claim 17, and as discussed above, claim 17 is not obvious (missing claimed element and no motivation to

combine) and therefore claim 21 is not obvious. Similarly, claim 26 depends from claim 22 and as discussed above, claim 22 is not obvious (missing claimed element and no motivation to combine) and therefore claim 26 is not obvious. In addition to the combination of McKeeman, Doo, and Eto not disclosing the claimed elements in either claim 17 or claim 22, the combination of references does not recite the additional elements claimed in claims 21 and 26. In claim 21, the additional element of using a generated dependency information to identify dependent objects that are invalid in a database is claimed. While Eto discusses identifying invalid DML statements in a transaction, Eto does not teach identifying invalid stored code objects. Therefore, the combination of McKeeman, Doo and Eto does not render claim 21 obvious and applicants respectfully request that this rejection be withdrawn. Furthermore, there is no teaching, motivation or suggestion to combine the references in the proposed manner provided in either the Office Action or the references. For this additional reason, applicants respectfully request that the rejection be withdrawn.

Similarly, concerning claim 26, Eto does not disclose identifying dependent objects that are invalid, particularly not for object oriented code object implementations stored in a database. Therefore, the combination of McKeeman, Doo and Eto does not render claim 26 obvious and applicants respectfully request that this rejection be withdrawn. Furthermore, there is no teaching, motivation or suggestion to combine the references in the proposed manner provided in either the Office Action or the references. For this additional reason, applicants respectfully request that the rejection be withdrawn.

Conclusion

Based on the foregoing remarks, the applicants believe that all of the claims in this case are in a condition for allowance and an indication to that effect is earnestly solicited. Furthermore, if the Examiner believes that additional discussions or information might advance the prosecution of this case, the Examiner should feel free to contact the undersigned at the telephone number indicated below.

Respectfully submitted,

By: Tara A. Kastelic

Tara A. Kastelic, Reg. No. 35,980
(Cust. No. 24024)
800 Superior Avenue - Suite 1400
Cleveland, Ohio 44114-2688
Phone (216) 622-8329
Fax (216) 241-0816

TAK0013